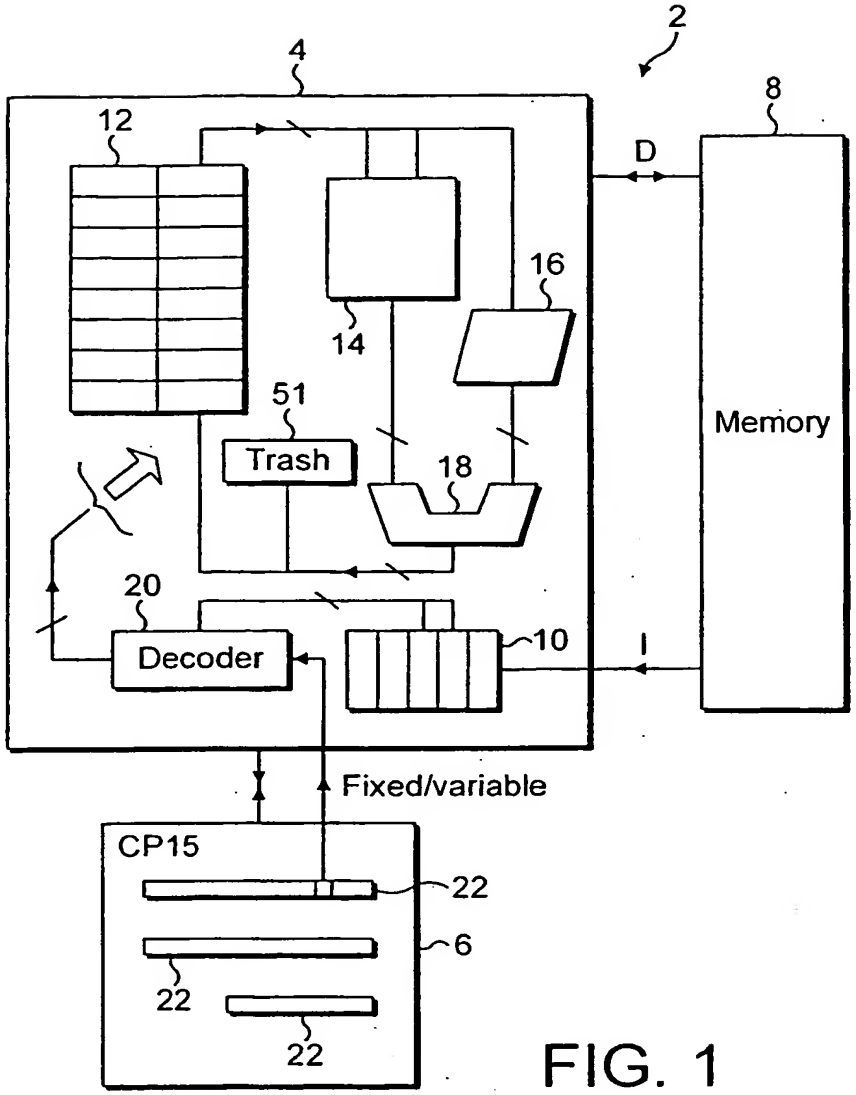

2
4
8
12
D
16
14
51
Trash
18
Memory
20
Decoder
10
I
Fixed/variable
CP15
22
6
22
22

FIG. 1

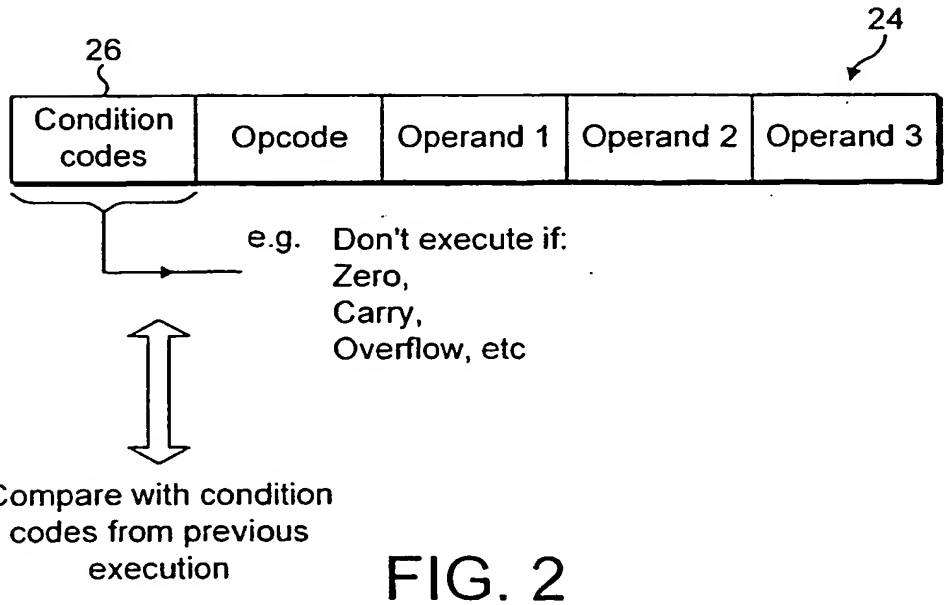

24
26
Condition codes
Opcode
Operand 1
Operand 2
Operand 3
e.g. Don't execute if:
Zero,
Carry,
Overflow, etc
Compare with condition codes from previous execution

FIG. 2

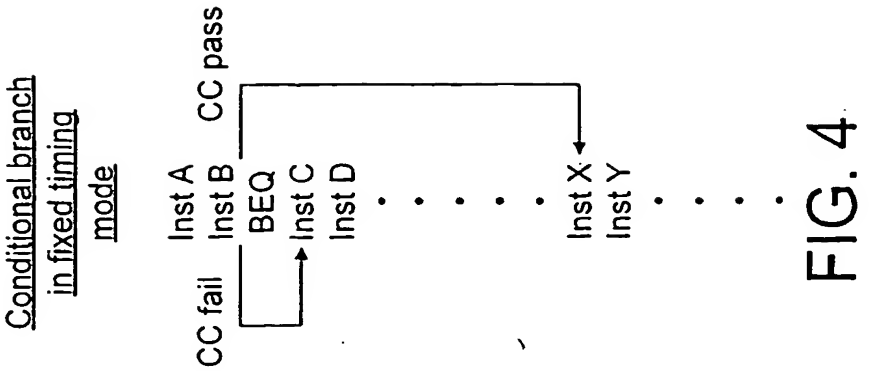

Conditional branch in fixed timing mode
Inst A
Inst B
BEQ
Inst C
Inst D
Inst X
Inst Y
CC pass
CC fail

FIG. 4

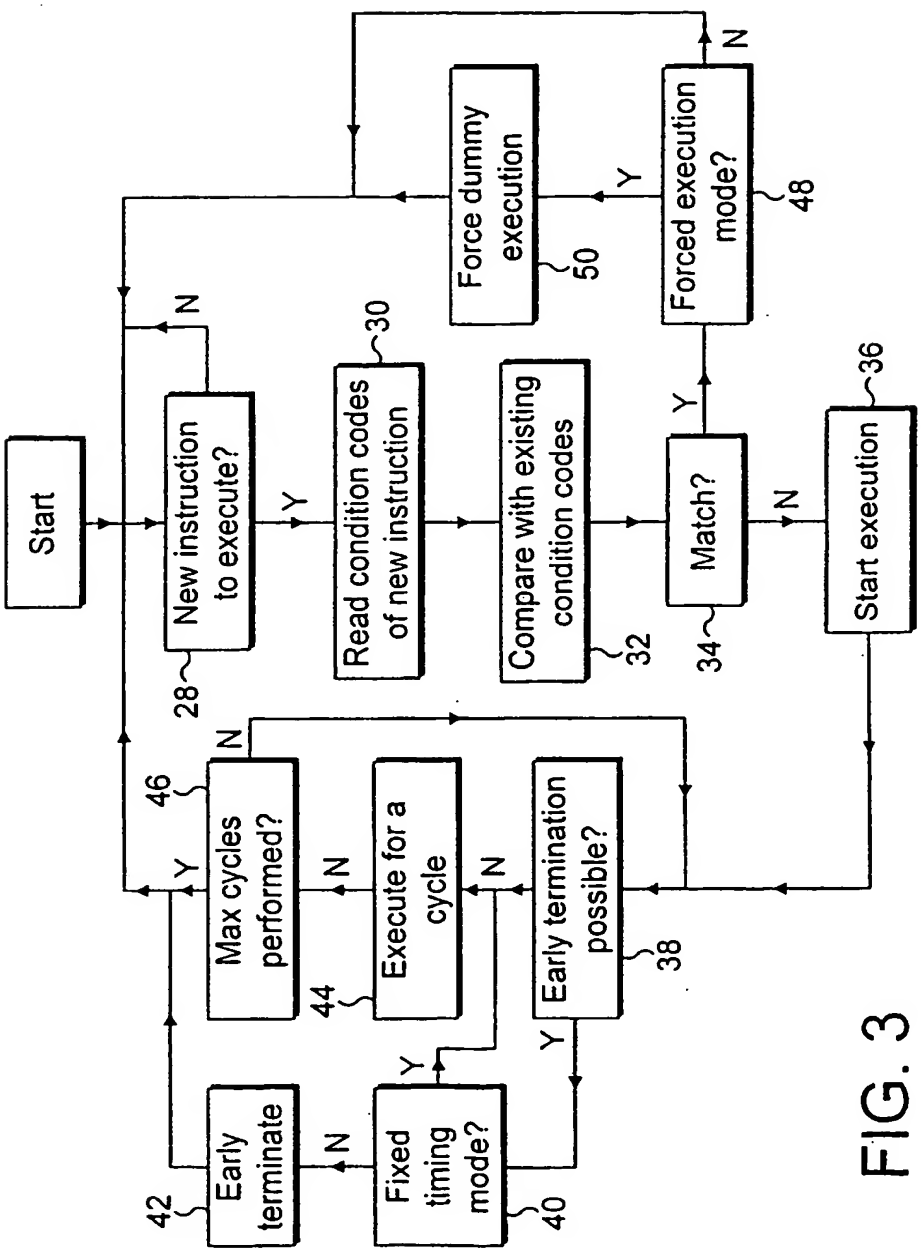

Start
New instruction to execute?
28
Read condition codes of new instruction
30
Compare with existing condition codes
32
Match?
34
Start execution
36
Force dummy execution
50
Forced execution mode?
48
Max cycles performed?
46
Execute for a cycle
44
Early termination possible?
38
Early terminate
42
Fixed timing mode?
40
Y
N

FIG. 3

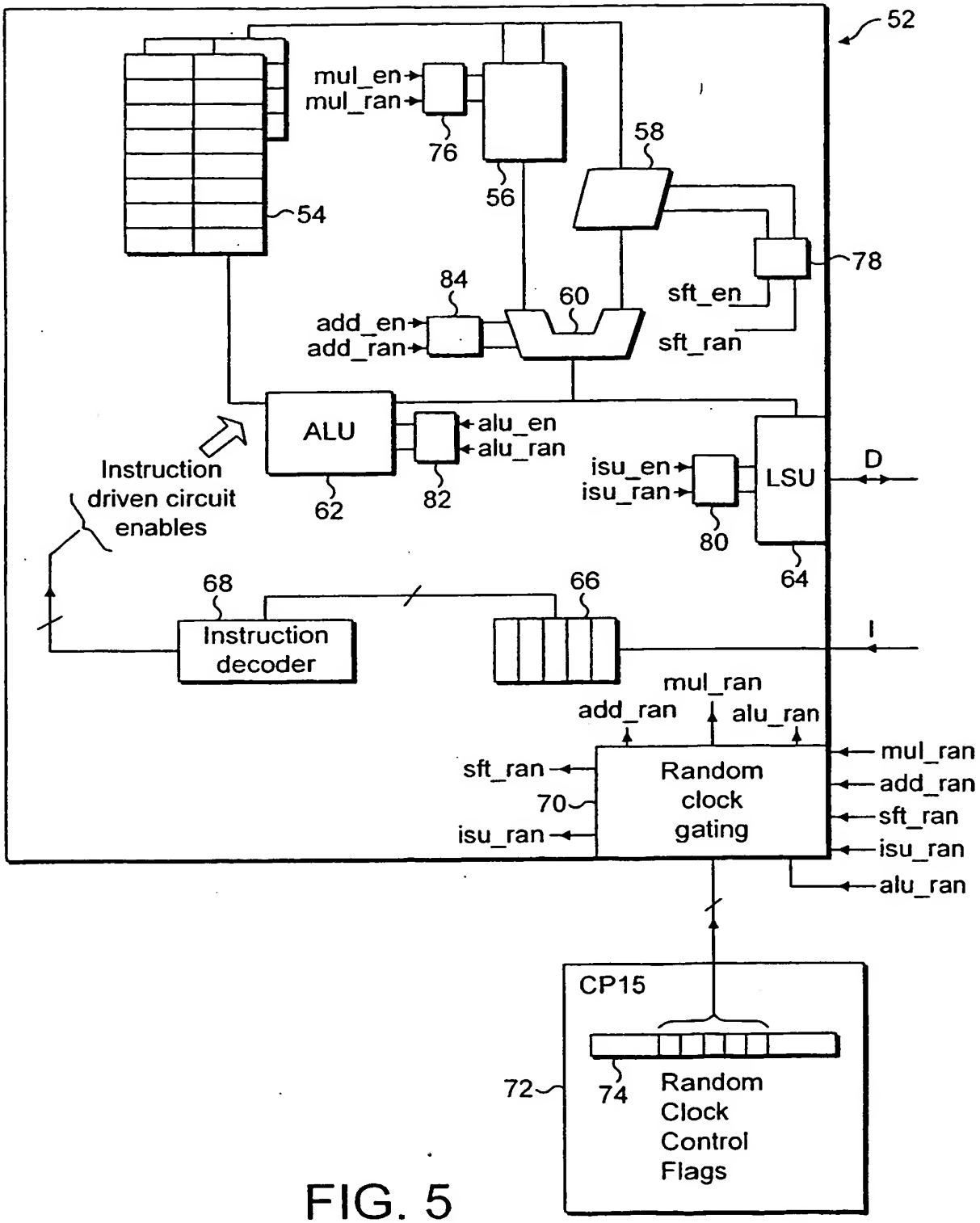

52
mul_en
mul_ran
76
56
58
54
78
84
60
sft_en
add_en
add_ran
sft_ran
ALU
alu_en
alu_ran
Instruction
driven circuit
enables
62
82
isu_en
isu_ran
LSU
D
80
64
68
66
Instruction
decoder
I
mul_ran
add_ran
alu_ran
sft_ran
70
isu_ran
Random
clock
gating
mul_ran
add_ran
sft_ran
isu_ran
alu_ran
CP15
72
74
Random
Clock
Control
Flags

FIG. 5

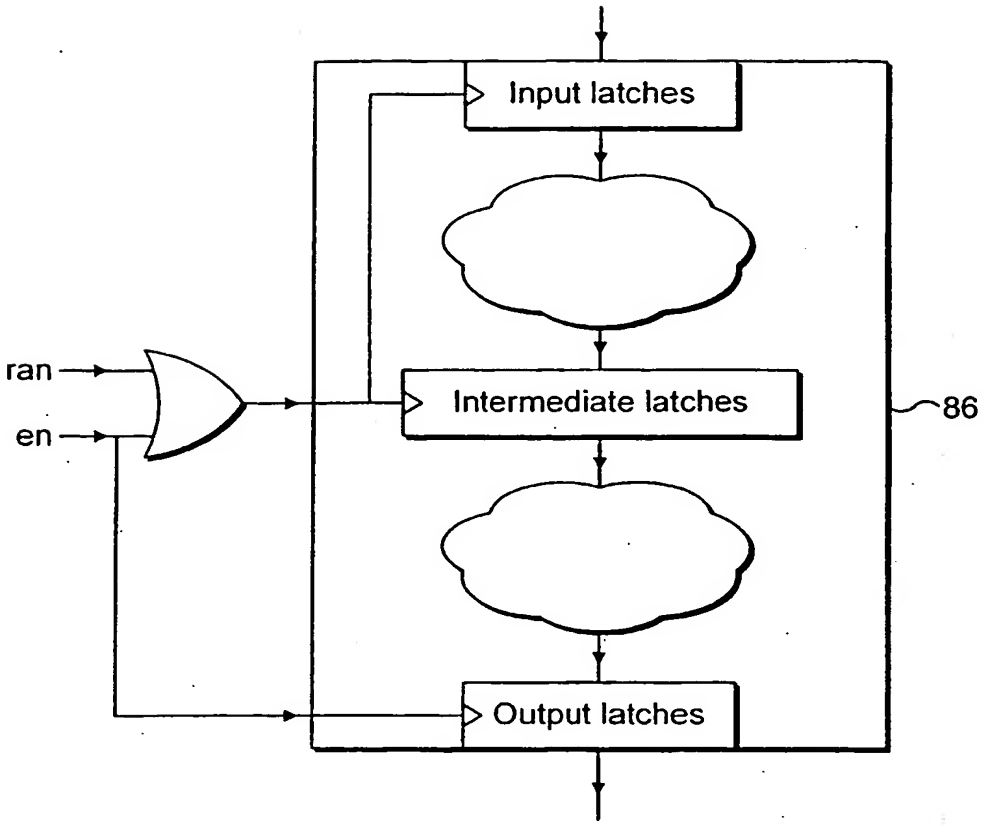

Input latches
ran
en
Intermediate latches
86
Output latches

FIG. 6

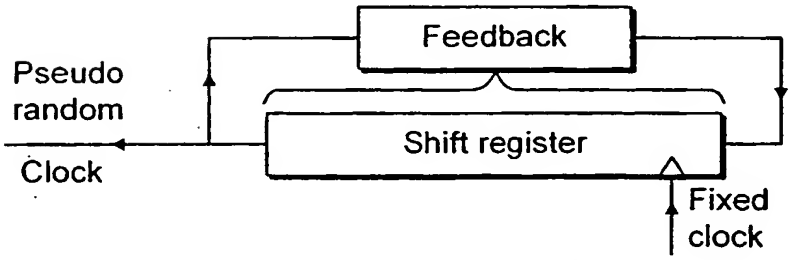

Feedback
Pseudo
random
Clock
Shift register
Fixed
clock

FIG. 7

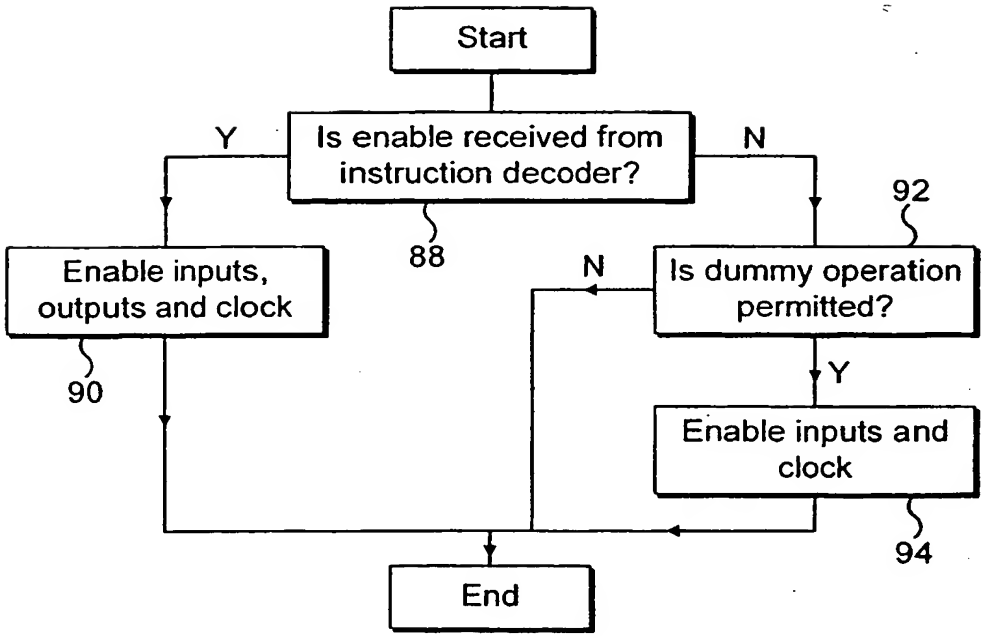

Start
Is enable received from instruction decoder?
Y
N
88
92
Enable inputs, outputs and clock
90
N
Is dummy operation permitted?
Y
Enable inputs and clock
94
End

FIG. 8

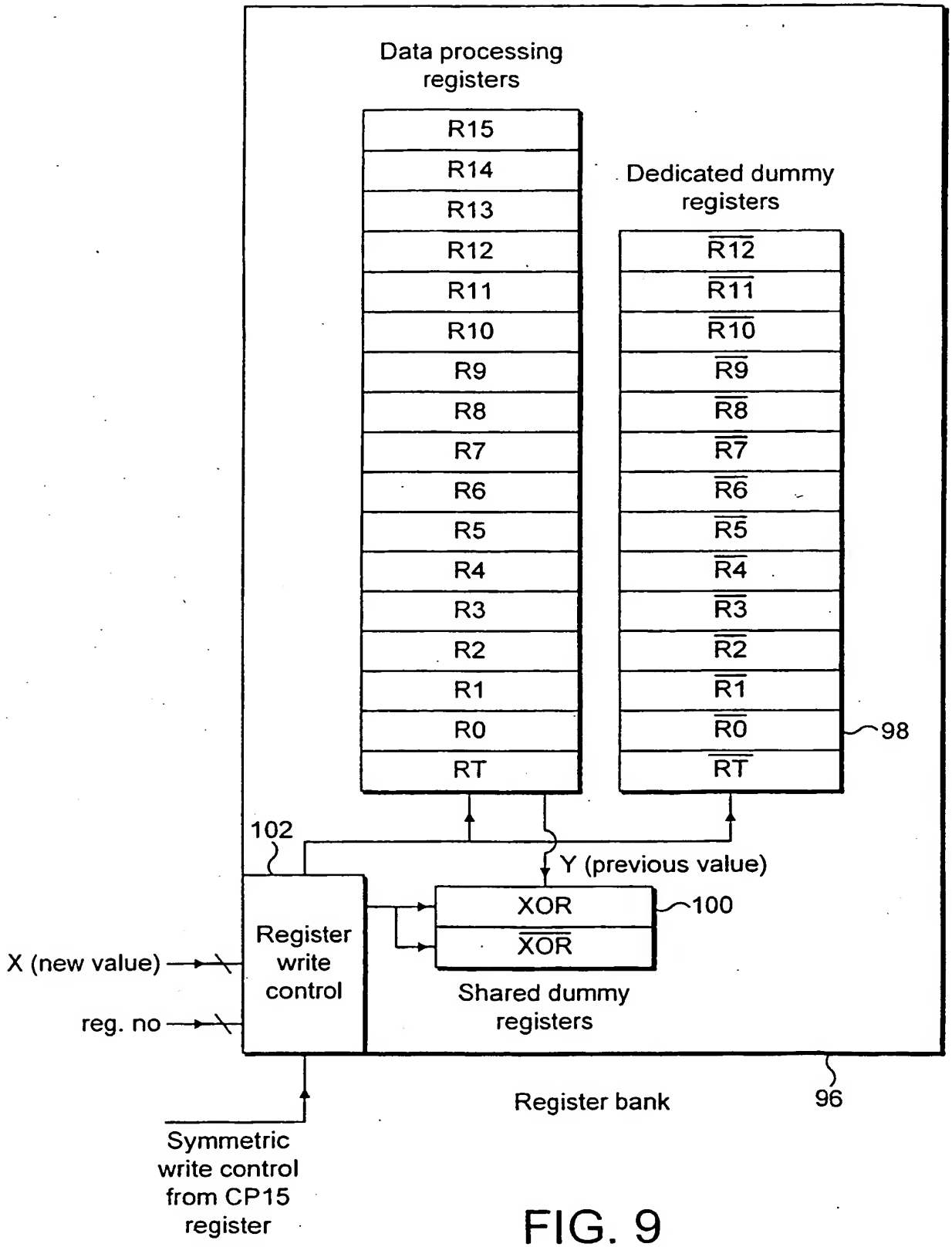

Data processing registers
R15
R14
R13
R12
R11
R10
R9
R8
R7
R6
R5
R4
R3
R2
R1
R0
RT
Dedicated dummy registers
$\overline{R12}$
$\overline{R11}$
$\overline{R10}$
$\overline{R9}$
$\overline{R8}$
$\overline{R7}$
$\overline{R6}$
$\overline{R5}$
$\overline{R4}$
$\overline{R3}$
$\overline{R2}$
$\overline{R1}$
$\overline{R0}$
$\overline{RT}$
98
102
Y (previous value)
XOR
$\overline{XOR}$
100
Register write control
X (new value)
reg. no
Shared dummy registers
Register bank
96
Symmetric write control from CP15 register

FIG. 9

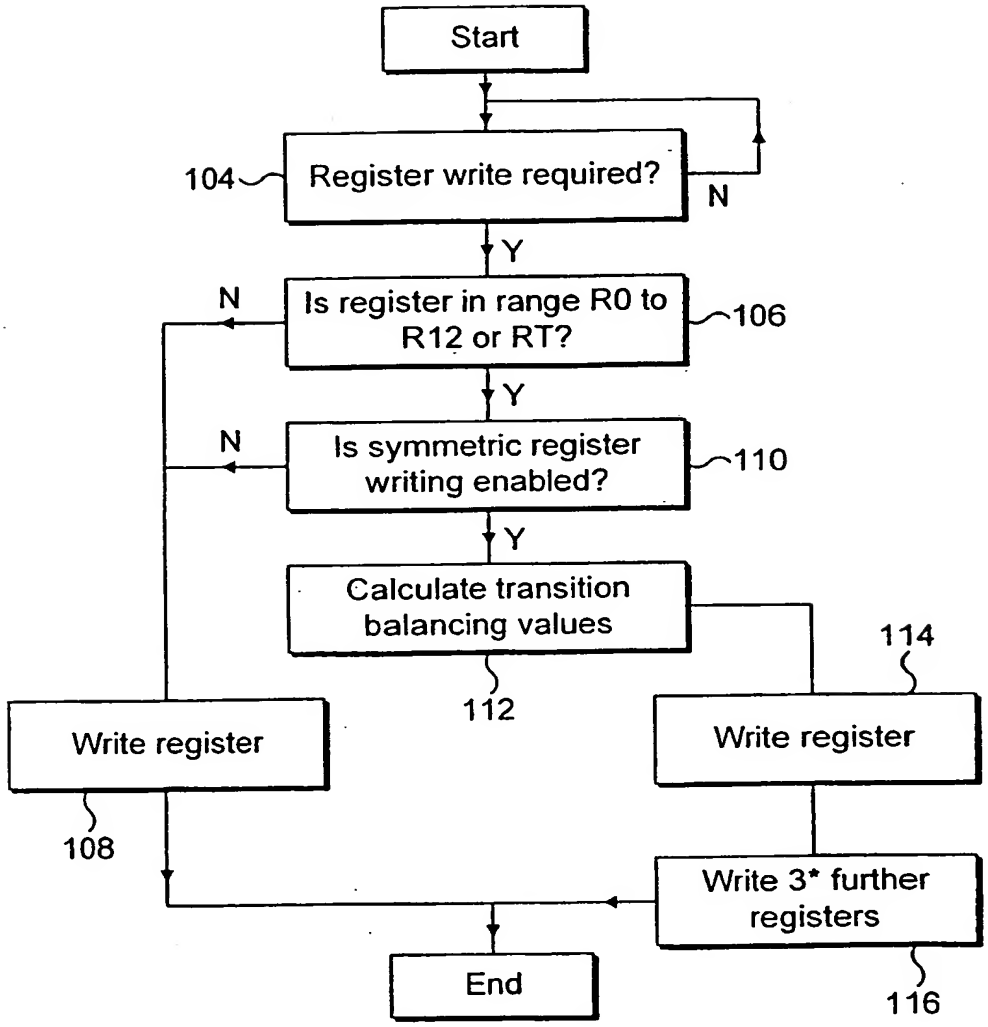
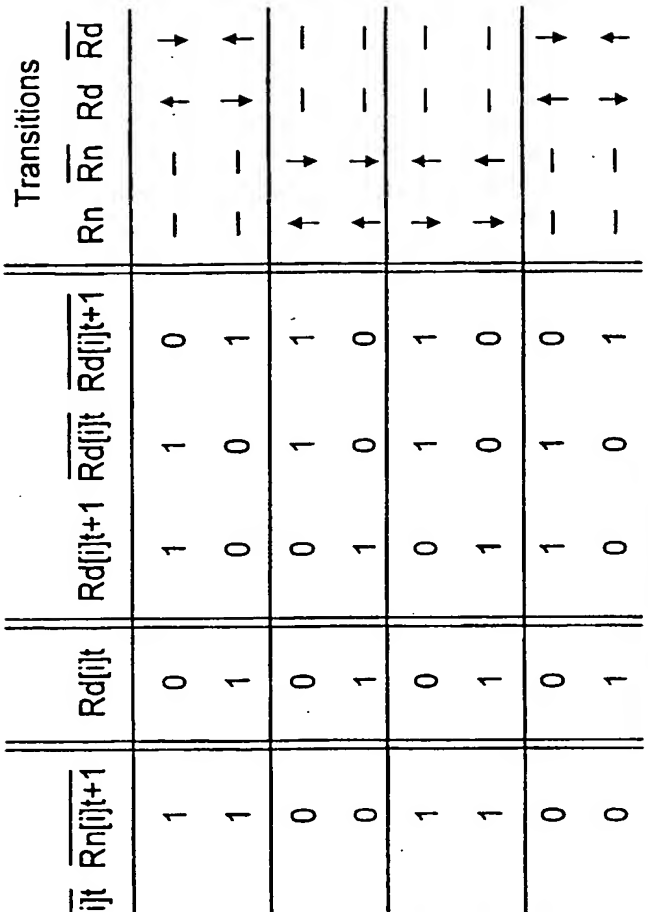

Start
Register write required?
104
N
Y
Is register in range R0 to R12 or RT?
106
N
Y
Is symmetric register writing enabled?
110
N
Y
Calculate transition balancing values
112
114
Write register
Write register
108
Write 3* further registers
116
End

FIG. 10



Rn[i]t	Rn[i]t+1	$\overline{Rn[i]t}$	$\overline{Rn[i]t+1}$	Rd[i]t	Rd[i]t+1	$\overline{Rd[i]t}$	$\overline{Rd[i]t+1}$	Transitions Rn	$\overline{Rn}$	Rd	$\overline{Rd}$
0	0	1	1	0	1	1	0	—	↓	↑	—
0	0	1	1	1	0	0	1	—	↓	↓	—
0	1	1	0	0	0	1	1	↑	—	—	↑
0	1	1	0	1	1	0	0	↑	—	—	↑
1	0	0	1	0	0	1	1	↓	—	—	↓
1	0	0	1	1	1	0	0	↓	—	—	↓
1	1	0	0	0	1	1	0	—	↓	↑	—
1	1	0	0	1	0	0	1	—	↓	↓	—

$$Rd[i]t+1 = \left(\overline{Rn[i]t \text{ XOR } Rn[i]t+1}\right) \text{ XOR } Rd[i]t$$

FIG. 11

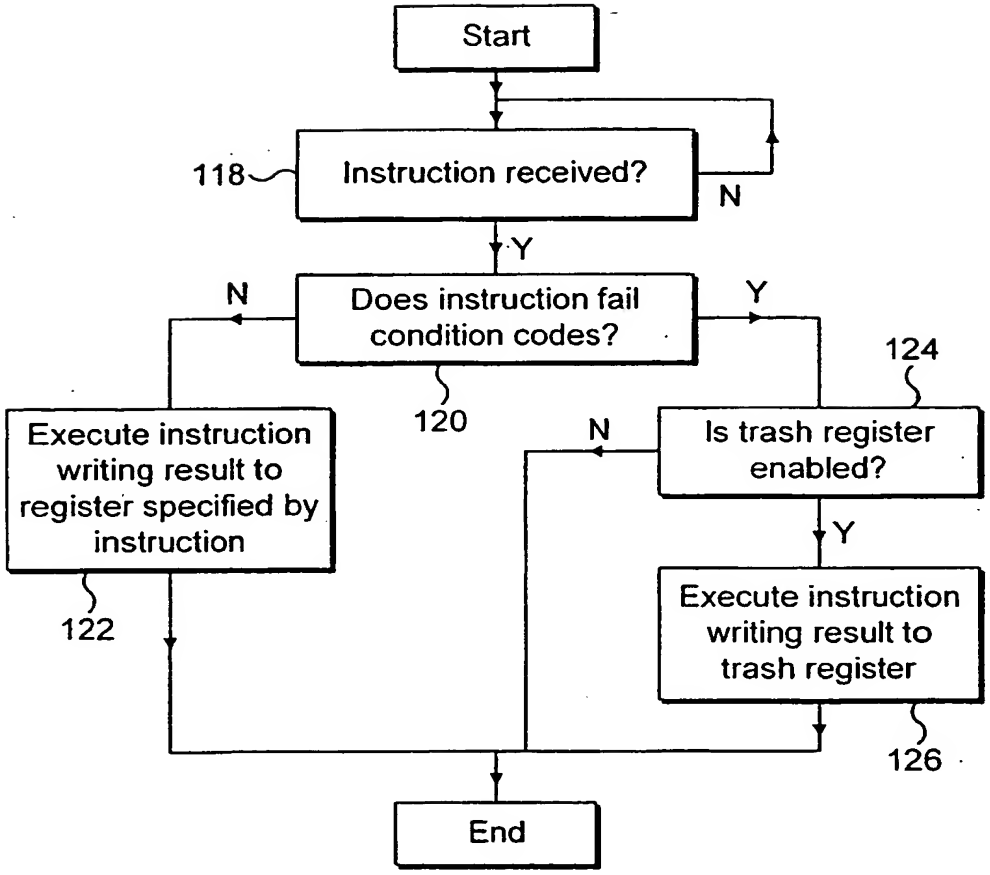

Start
Instruction received?
118
N
Y
Does instruction fail condition codes?
N
Y
120
124
Execute instruction writing result to register specified by instruction
122
N
Is trash register enabled?
Y
Execute instruction writing result to trash register
126
End

FIG. 12

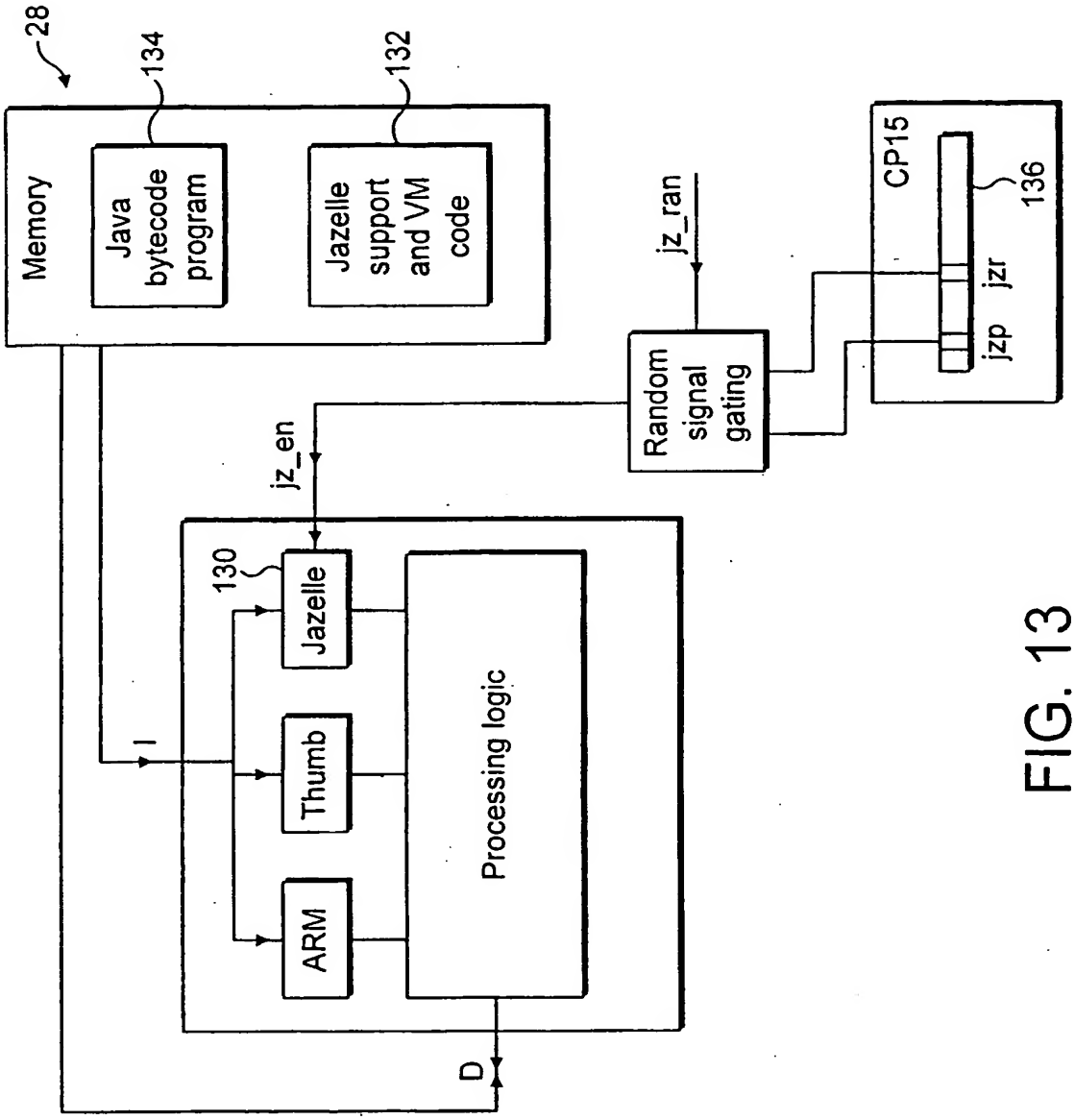

28
134
132
Memory
Java bytecode program
Jazelle support and VM code
jz_ran
CP15
136
jzr
jzp
Random signal gating
jz_en
130
Jazelle
Thumb
ARM
Processing logic
I
D

FIG. 13

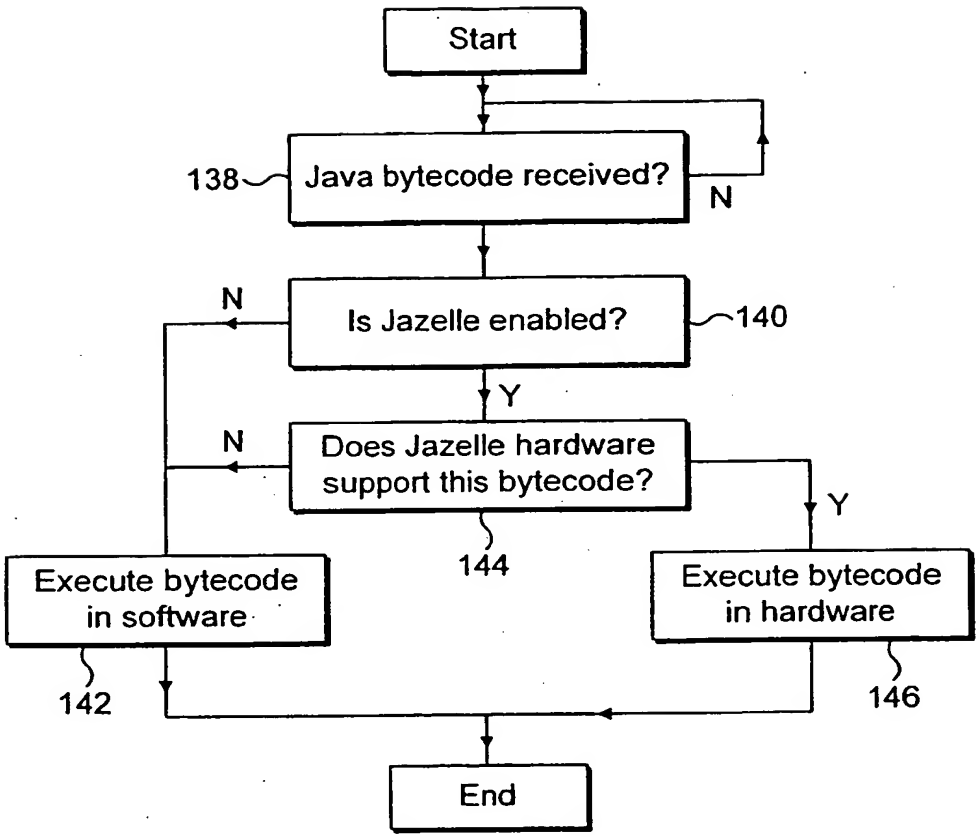

Start
138
Java bytecode received?
N
N
Is Jazelle enabled?
140
Y
N
Does Jazelle hardware support this bytecode?
Y
144
Execute bytecode in software
Execute bytecode in hardware
142
146
End

FIG. 14